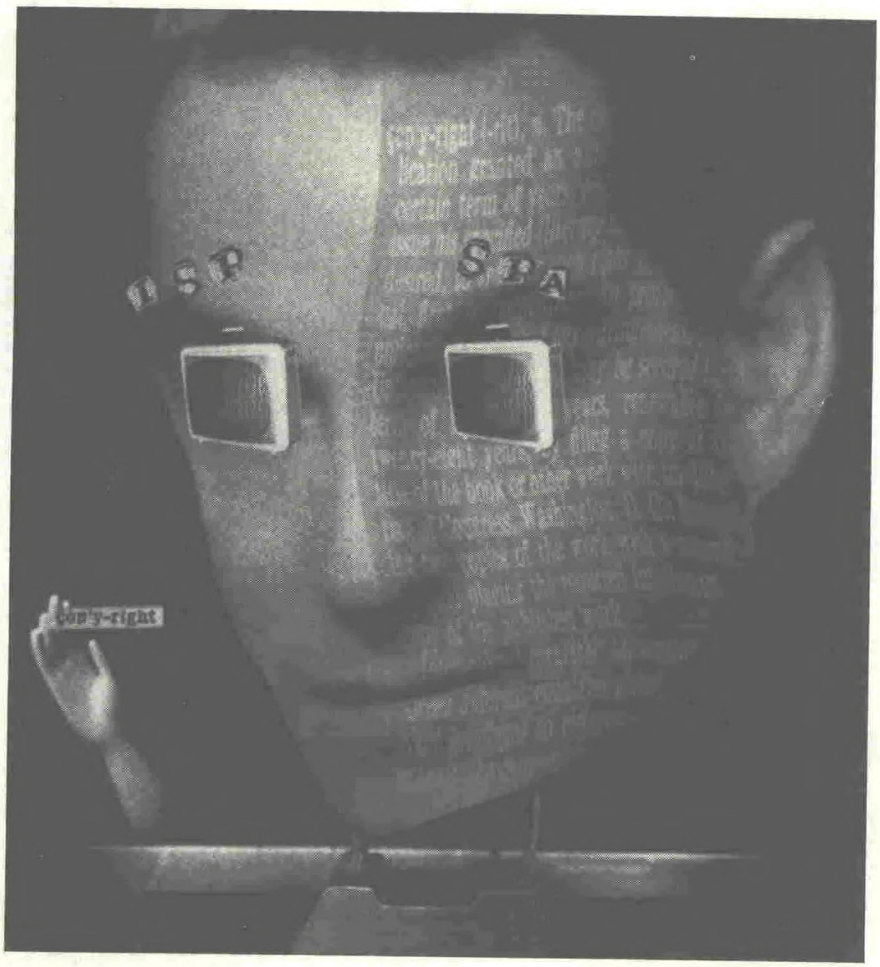
ยืน ภู่วรวรรณ

ปี 2000
ปัญหาที่เกิดกับคอมพิวเตอร์

มาดูกันว่าปัญหาที่แท้จริงอยู่ที่ใด ปัญหาเหล่านั้นมีความเป็นมาอย่างไร มูลเหตุแห่งปัญหา และก็คงหาวิธีการว่า มีวิธีการใดในการแก้ปัญหา



ในรอบปีที่ผ่านมา เราได้ยิน ได้ฟัง ได้เห็นจากข่าวหนังสือพิมพ์บ้าง ทางโทรทัศน์บ้าง ต่างก็ว่าปัญหาปี 2000 จะทำให้คอมพิวเตอร์หยุดทำงาน หรือทำงานผิดพลาด จนกลายเป็นข่าวใหญ่โตในรอบปี

ผู้เขียนพยายามตรึกตรองดูว่า ปัญหาที่เกิดขึ้นจะเป็นจริงดังที่มีข่าวหรือไม่ หลังจากไตร่ตรองอยู่นาน จึงคิดว่าน่าจะมาดูกันว่าปัญหาที่แท้จริงอยู่ที่ใด ปัญหาเหล่านั้นมีความเป็นมาอย่างไร มูลเหตุแห่งปัญหา และก็คงหาวิธีการว่า มีวิธีการใดในการแก้ปัญหา

ก่อนอื่นผู้เขียนต้องออกตัวสักเล็กน้อยว่า ความลึกซึ้งของผลกระทบที่จะเข้าไปอยู่ในตัวโอเอส หรือโปรแกรมระบบต่าง ๆ เป็นอย่างไรนั้น ผู้เขียนไม่อาจทราบได้ ปัญหาที่หลายคนว่าเป็นเรื่องสาหัสนั้น จึงอาจมองไม่เห็นชัดนัก แต่อย่างไรก็ตาม ผู้เขียนพยายามเขียนจากภูมิความรู้ที่มีอยู่ และวิเคราะห์หาปัญหา

ต้นเหตุแห่งปัญหาพอสรุปได้ว่า เกิดจากการที่โปรแกรมคำนวณวันที่ใช้ รูปแบบ MMDDYY หรือใช้ตัวย่อปีเพียงสองหลัก ทำให้ในปี ค.ศ. 2000 ตัวเลขปีกลับมาที่ 00 หรือมีค่าเป็น 0 ซึ่งซ้ำกับปี 1900 อาจทำให้เกิดการคำนวณบางอย่างผิดพลาด เพราะโปรแกรมที่เขียนไว้อาจทำงานผิด

ต้นเหตุของปัญหาอยู่ที่ ปี ค.ศ. ใช้สองหลัก

ปัญหานี้เป็นปัญหาที่น่าจะมองเห็นแล้วตั้ง 20 ก่อน ในยุคที่คอมพิวเตอร์เมนเฟรมกำลังใช้งานกันอยู่ และจำนวนเครื่องคอมพิวเตอร์ก็ยังไม่มากนัก การกำหนดรูปแบบของ

```
B>type dayweek.prg
*time is money....you see!
*this program' I will show you how to calculate
*the day of week and julian date
*and you can use this program to calculate
*how many days have elapsed between any date.
*ok!
*
*copy wrong Sept, 1985
*
*Parameters passed
*     name          type          len    picture      description
      ----          ----          ---    -------      -----------
* in: mdate         C             10     MM/DD/YYYY   Calender date
* out:is:valid      L             1                   Flag
*     julian        N             10                  julian date
*     weekday       c             9                   day of were
                                                      string

*
*
set talk off
store val ($(mdate,1,2)) to month
store val ($(mdate,4,2)) to day
store val ($(mdate,7,4)) to year
store T to is:valid
store o to julian
store [ ] to weekday
*
*
if    month<1.or.month>12.or.day<1.or.;
    day>val($("312931303130313130313031",;
    (month-12*int (month/12))*2-1,2)).or.;
    (month=2.and.day>28.and.((year/4.0>int(year/4.0);
    .and.year/100.0=int (year/100.0)).or.;
     year/400.0>int (year/400.0)))
           store F to is:valid
           release month,day,year
           return
endif
*
*Julian date calculation from Ashton-Tate demo
*
store val($(mdate,7,2)) to century
store val ($(mdate,9,2)) to yearcount
store int (30.57*month)+int(365.25*year-395.25);
      +day to julian
if month > 2
     if year/4.0=int(year/4.0)
      store julian-1 to julian
     else
      store julian-2 to julian
     endif
      store month-2 to month
else
     store month+10 to month
     store yearcount-1 to yearcount
endif
*
*
store int((13*month-1)/5)+day+yearcount+;
      int (yearcount/4)+int(century/4)-2*century to mday
store int (mday-7*int (mday/7)) to mday
store $ ("SUNDAY MONDAY TUESDAY WEDNESDAY THURSDAY ;
FRIDAY SATURDAY ",mday*9+1,9) to weekday
*
*test the result
?   "julian date = ",julian
?   "weekday = "+ weekday
return
```

รูปที่ 1 โปรแกรมการคำนวณเกี่ยวกับวันที่และวันในสัปดาห์

ปี ค.ศ. โดยใช้ MMDDYY หรือยอมรับใช้ปี ค.ศ. เพียงสองหลักนั้น เป็นการสร้างวัฒนธรรมที่ไม่ถูกต้องนักในการเขียนโปรแกรม เพราะหากเขียนโปรแกรมจะต้องยอมรับให้ผู้ใช้ป้อนข้อมูลแบบสองหลักได้ แต่การเก็บข้อมูลภายในทั้งหมดต้องเก็บแบบเต็ม

การคำนวณวันที่ตามสูตรการคำนวณจำเป็นจะต้องอ้างอิงกับวันใดวันหนึ่ง โดยทั่วไปเรามักใช้คำจูเลียน คือวันที่นับอ้างอิงกับวันที่ 1 มกราคม ค.ศ. 1 ผู้เขียนเคยเขียนโปรแกรมการคำนวณวันที่เพื่อหาค่าจูเลียนและนำมาใช้ในการบวกลบ การคิดคำนวณถ้าอ้างอิงจากวันที่ 1 มกราคม ค.ศ. 1 จนถึงวันที่คิดคำนวณเพื่อให้ได้จำนวนวัน

ลองดูตัวอย่างโปรแกรมที่ผู้เขียนเขียนไว้ในวารสารไมโครคอมพิวเตอร์ ฉบับที่ 17 เดือนพฤศจิกายน 2528 โดยสามารถนำมาใช้ในการหาจำนวนวันและวันที่ในสัปดาห์ได้

อย่างไรก็ดี หากย้อนไปดูประวัติการเกิดปฏิทินกิกอเลียน ที่ถือว่าเป็นปฏิทินที่ใช้กันในปัจจุบันและเป็นปฏิทินที่เกิดภายหลังเมื่อไม่กี่ร้อยปีมานี้ การปรับแต่งวันให้มีปีอธิกสุรทิน (กุมภาพันธ์มี 29 วัน) เพื่อชดเชยเวลา เพราะการหมุนของโลกครบหนึ่งรอบดวงอาทิตย์ใช้เวลาประมาณ 365.25 แต่ไม่ใช้ .25 พอดี ดังนั้นปีอธิกสุรทินที่ชดเชยจึงไม่ใช้ทุก 4 ปี แต่จะมีการปรับแต่งในปีที่ลงท้าย 00 หรือทุกร้อยปีและ 400 ปี ดังตัวอย่างสูตรการคำนวณที่เขียนไว้ในโปรแกรม กล่าวคือ ปีที่ลงท้าย 00 จะมี 28 วันบ้าง 29 วันบ้าง เป็นที่โชคดีที่ปี ค.ศ. 2000 มี 29 วัน

จะเห็นว่าหากการคำนวณคิดตามหลักการนี้คือใช้ค่าจูเลียนที่เริ่มจากปี ค.ศ. 1 ปัญหาจะไม่เกิดขึ้น แต่การคำนวณในหลายโปรแกรมแม้แต่ในโปรแกรมสเปรดชีตคิดการคำนวณค่าจูเลียน โดยอ้างอิงกับวันที่ 1 มกราคม ปี ค.ศ. 1900 เพื่อให้ตัวเลขน้อยลง ความจริงค่านี้ก็ไม่น่ามีปัญหา ปัญหาอยู่ที่ว่าอ้างอิงกับตัวเลข 2 หลักสุดท้ายหรือทั้งหมด ถ้าอ้างอิงกับสองหลักสุดท้าย ปี ค.ศ. 2000 ก็จะกลายเป็น 100 ซึ่งก็จะไม่มีปัญหา แต่หากปี ค.ศ. 2000 เป็นตัวเลข 00 ก็จะเกิดปัญหา

รูปที่ 2
การแสดงผล
ของโปรแกรม
คำนวณวันที่
และวันในสัปดาห์

```
. store "08/25/1985" to mdate
08/25/1985
. do dayweek
julian date =            724893
weekday = SUNDAY
.
.
.
. store "10/18/1985" to mdate
. do dayweek
julian date =            724947
weekday = FRIDAY
.
.
.
. stroe "11/05/1950" to mdate
. do dayweek
julian date =            712181
weekday = SUNDAY
```

ปัญหา จากการเขียนโปรแกรม

จากที่กล่าวมาแล้วในหัวข้อก่อนจึงพอบอกได้ว่า ปัญหาที่แท้จริงเกิดจากการที่ผู้เขียนโปรแกรมยอมรับให้มีการป้อนข้อมูลวันที่แบบใช้ ค.ศ. สองหลักและเก็บข้อมูลไว้ใช้ในการคำนวณแบบ 2 หลัก โดยเน้นเป็นที่เข้าใจว่า 2 หลักต้นคือ 19

การเขียนโปรแกรมเหล่านี้เริ่มจากการกำหนดค่าวันที่ในระบบโอเอส เพราะโอเอสทุกเครื่องจะมีปฏิทินของตัวเอง การตั้งค่าและการคำนวณค่าในเรื่องเวลาทำได้ตั้งแต่ในโปรแกรมระบบ ปัญหาประเด็นนี้ผู้พัฒนาโอเอสจะต้องตามแก้ไข ซึ่งตามความเห็นของผู้เขียนก็ไม่น่ามีปัญหา เพราะเป็นเรื่องที่แก้ไขกันได้อยู่แล้ว และโอเอสใหม่ ๆ ก็คงได้เตรียมการเรื่องนี้อยู่แล้ว ความรับผิดชอบในการแก้ไขโปรแกรมจึงอยู่ที่ผู้พัฒนาโอเอส

ปัญหาอยู่ที่การใช้โปรแกรมประยุกต์โดยเฉพาะโปรแกรมประยุกต์เก่าที่ใช้กันมานานแล้ว บางโปรแกรมเขียนด้วยภาษาโคบอล ซึ่งมีรูปแบบของการเขียนวันที่เป็นแบบ YY ได้ ทำให้การสร้างข้อมูลภายในเพื่อใช้งานของตนเองเป็นแบบตัวเลข 2 ตัว เช่น ในระบบบัญชีที่มีการลงวันที่ ก็จะเก็บไว้เพียงแค่ 2 หลัก ในกรณีนี้อาจมีปัญหาแน่นอน เพราะโปรแกรมที่คิดคำนวณอาจคำนวณผิดพลาดได้เมื่อถึง ค.ศ. 2000 อย่างไรก็ตาม การแก้ไขโปรแกรมเหล่านี้ก็ย่อมทำได้ แต่มีผู้สันนิษฐานว่าหลายโปรแกรมที่เคยเขียนด้วยโคบอลและใช้กันมานานจนทำให้ยากต่อการแก้ไข และการเขียนโปรแกรมเหล่านี้อาจมีโปรแกรมหลายร้อยหลายพันโปรแกรมทำให้สิ้นเปลืองในการแก้ปัญหา

สำหรับเครื่องคอมพิวเตอร์ส่วนตัวหรือการใช้โปรแกรมสำเร็จรูปทั่วไป ปัญหาเหล่านี้ดูจะน้อยลง เพราะการใช้งานส่วนใหญ่เป็นงานเฉพาะตัว และโปรแกรมสำเร็จรูปเหล่านี้ก็พัฒนาเวอร์ชันใหม่อยู่ตลอดเวลา ปัญหาในประเด็นเหล่านี้คงไม่น่าจะเป็นปัญหา

ข้อควรระวัง สำหรับปัญหาที่เกิดขึ้น

จากที่เราทราบปัญหาแล้วว่า หากเราใช้ ปี ค.ศ. เป็นข้อมูลในการใช้งาน และมีการนำเอามาใช้คิดคำนวณต่อ ดังนั้นการเก็บข้อมูลในรูปของปี ค.ศ. ทุกครั้งจะต้องเก็บเต็มหลัก หรือหากเก็บสองหลักต้องแน่ใจว่าสามารถจัดการได้เมื่อถึงปี ค.ศ. 2000 การเตรียมการตั้งแต่ปัจจุบันเป็นเรื่องที่ป้องกันปัญหาในอนาคตได้มาก หลายคนอาจคิดว่าปี ค.ศ. 2000 ยังอีกนาน แต่กาลเวลาผ่านไปอย่างรวดเร็ว

ขณะเดียวกัน ถ้ามีงานเดิมที่ใช้งานอยู่มีการใช้ปี ค.ศ. ในรูปตัวเลขสองหลัก จำเป็นต้องวางแผนการแก้ไข และหาทางดำเนินการแก้ไขเสียแต่เนิ่น ๆ การแก้ไขตั้งแต่วันนี้ก็คงจะลดปัญหาในอนาคตได้มาก อย่างไรก็ตาม หลายอย่างเราอาจไม่ทราบปัญหาได้ บางครั้งการแปลงข้อมูลหรือการดำเนินการบางอย่างเกี่ยวกับการแก้ไขจึงไม่เห็นสภาพชัดเจน สิ่งเหล่านี้อาจเกิดขึ้นจากการทดลองหรือทดสอบระบบงานของเรา โดยการลองสมมติไปที่เวลาของปี ค.ศ. 2000 หรือกว่าแล้ว จำลองดูว่ามีปัญหาใดเกิดขึ้นบ้าง แล้วจึงแก้ปัญหาต่าง ๆ เหล่านี้

หากองค์กรใดมีโปรแกรมเมอร์ หรือนักพัฒนาระบบ ก็คงต้องให้ผู้เกี่ยวข้องเหล่านั้นเข้าใจและตระหนักถึงปัญหาที่เกิดขึ้น ซักซ้อมความเข้าใจเพื่อจะได้เตรียมการแก้ปัญหาเหล่านี้ หรือไม่สร้างปัญหาให้กับระบบใหม่อีก

ปัญหา ไม่น่ากลัวอย่างที่เป็นข่าว

จากการประเมินสถานการณ์ของปัญหาพบว่า ปัญหาจากปี ค.ศ. 2000 ไม่น่าจะเป็นปัญหาใหญ่ เพราะเป็นปัญหาที่รู้ว่าอะไรคือต้นเหตุแห่งปัญหา เป็นปัญหาที่คนสร้างขึ้นมาและก็คงแก้ไขปัญหานี้ได้ อีกทั้งมีเวลาเตรียมการกันนาน ประจวบกับเทคโนโลยีคอมพิวเตอร์มีอายุสั้นลงมาก โปรแกรมจำนวนมากก็ล้าสมัยลง มีการปรับปรุงเวอร์ชันใหม่ตลอดเวลา หรือแม้แต่โปรแกรมที่องค์กรพัฒนากันเองก็มีการปรับปรุงอยู่ตลอดเวลาแล้ว การปรับปรุงสิ่งต่าง ๆ เหล่านี้เป็นสิ่งที่เป็นไปได้ และจะกระทำได้โดยทันการณ์

ผู้เขียนมีความเชื่อว่าปัญหานี้เป็นปัญหาที่ไม่น่าหนักใจอะไรมากนัก เพราะต้นเหตุของปัญหาชัดเจน ผิดกับบางปัญหาเช่นปัญหาเรื่องไวรัสคอมพิวเตอร์ ที่ต้นเหตุของปัญหาคือพวกโรคจิตพัฒนาหลบซ่อนและไม่เปิดเผย ปัญหาเรื่องความปลอดภัยของข้อมูลบนเครือข่ายและการโจรกรรมข้อมูล ซึ่งปัญหาเหล่านี้เป็นปัญหาที่ไม่เปิดเผย ดังนั้นจึงยากที่จะรับรู้ได้

อย่างไรก็ดี เราก็คงไม่อยากพบกับปัญหาใด ๆ ดังนั้นจากสุภาษิตที่ว่า "กันไว้ดีกว่าแก้ แย่แล้วแก้ไม่ทัน" ก็น่าจะใช้ได้จึงอยากให้ทุกท่านที่เกี่ยวข้องหาทางป้องกันปัญหาไว้ก่อน